PERFORMANCE ANALYSIS OF CIRCUIT SWITCHING BASELINE INTERCONNECTION NETWORKS

Manjai Lee and Chuan-lin Wu

Department of Electrical Engineering
University of Texas at Austin
Austin, TX 78712

Abstract

Performance evaluation, using both analytical and simulation models, of circuit switching baseline networks is presented. Two configurations of the baseline networks, single and dual, are evaluated. In each configuration, two different conflict resolution strategies, drop and hold, are tried to see the performance difference. Our analytical models are based on a more realistic assumption. New analyses are given and are verified by simulation results. In single network configuration, it is shown that the drop strategy is better than the hold strategy in the case that the data transfer time is longer than 10 cycles under a high request rate. In the dual network configuration, five different communication strategies are investigated and the optimum performance level is shown to be dependent on the length of the data transfer time.

I. Introduction

Multistage interconnection networks have been proposed by many research groups for interconnecting multiple processors [1]. Performance evaluation work on the networks has also been done quite extensively [2-5]. However, previous evaluation models are commonly based on the unrealistic assumption that a blocked request is discarded and an independent request is generated to replace the previously blocked and yet unserved request. This assumption helps researchers simplify the theoretical model, but the simplification will result in discrepancies in predicting network performance.

In addition, there is no quantitative measure existing on how to choose a way to handle the blocked request between the two alternatives: hold and drop. In the hold strategy, the blocked request holds the partial path already established and waits for a release of the blockage. In the drop strategy, the blocked request abandons the partial path already established and starts over again. A quantitative measure on each strategy will provide insight information on the switching element design. Furthermore, most previous works consider only a single interconnection network. However, more than one network can be used in a system to enhance the performance. It is desirable to know how multiple networks can enhance the performance.

In this paper trying to solve the above problems, we formulate new models of circuit switching baseline interconnection networks [6,7]. The rest of the paper is organized as follows. Section II describes the network organizations which we will analyze. Assumptions for analytical model are presented in section III. For comparison, the regeneration model is described in section III, which uses the assumption that a blocked request is discarded and an independent request is generated. Sections IV and V present new analytical models on the hold and drop strategies respectively. Simulation and numerical results are provided in Section VI for comparison, followed by the conclusion.

II. Network Operations

The baseline network[6] is used here for study. The results obtained apply to other similar networks since topological equivalence has been proven [6]. A baseline network that has 8 input ports and 8 output ports is shown in Fig.1.

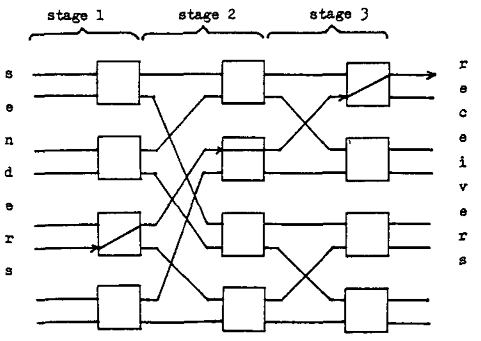

Fig. 1. 8×8 baseline network.

A processor that is connected to an input port and can generate and send requests to other processors is called a sender while a processor that is connected to an output port and receives requests is called a receiver. A circuit connection between a sender and a receiver is called a path. A 2×2 switching element is shown in Fig.2. A request from either one of the inputs can be connected to either one of the outputs if the output is not occupied by some other request. If the switching element connects an input to an output as requested, we say that the request is passed. A path between an input and output in a switching element is called a switching connection. When a request has arrived at the input of a switching element in stage i and if it has not passed the switching element, we say that the request is in stage i. In general, there are three possible states for a request at a switching element as shown in Fig.2.

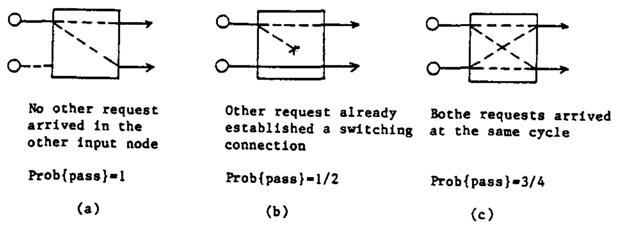
No other request arrived in the other input node
Prob{pass}=1
(a)
Other request already established a switching connection
Prob{pass}=1/2
(b)
Bothe requests arrived at the same cycle
Prob{pass}=3/4
(c)

Fig. 2. Probability of establishing a connection in a switching element.

In state (a), there is no previous request from the inputs and the probability for a new request to pass is equal to 1. In state (b), there is a request already passed, and probability for a new request to pass is equal to 1/2. State (c) has two requests arrival at the same cycle. If both requests ask for a same output, one will be passed while the other is blocked. The choice between the requests in the conflicting case depends on priority set in the switching element. The probability for a request to pass in state (c) is equal to 3/4 given that intended output for each request is equally distributed between both outputs.

When a sender generates a request, it is delivered through a link to an input of a switching element in stage 1. The request will pass stage 1 with probability of 1, 0.5, or 0.75 depending on the state of the other input of the switching element. Requests that pass stage 1 will progress to stage 2 and so on the same way. When a request passes stage n ($=\log_2 N$), where N is the number of senders or receivers a path is established between the sender and the receiver. A request can pass one stage at a cycle, therefore at least n cycles are required to establish a path between a sender and a receiver.

In the drop strategy, the switching element that has decided the blockage of a request sends back a release signal to the sender along the partially established path. This path is cleared at the end of the cycle. The blocked request, now dropped, goes under the same process starting from stage 1 until a path is established. In the hold strategy, the blocked request keeps the partial path and continues the connection effort at the same stage. The idea behind the drop strategy is to reduce the traffic of the requests in the network, which increases the probability of pass in the stages near to the receivers. But if a request has nearly made a path and has been dropped, there is much waste of effort. On the other hand, the idea behind the hold strategy is to save that kind of waste, but there is more tendency of a traffic jam than the drop strategy.

In addition to the single network configuration mentioned above, we also consider a dual network configuration as shown in Fig.3, which is made of two baseline networks connected side by side and used in parallel.

In the dual configuration, there is an additional multiplexor stage at the receiving side of the processor. A register in the multiplexor holds the information whether a receiver is busy receiving data from a sender. Due to this extra multiplexor stage, the minimum setup time of a path is one cycle longer than that of the single network with same number of senders.

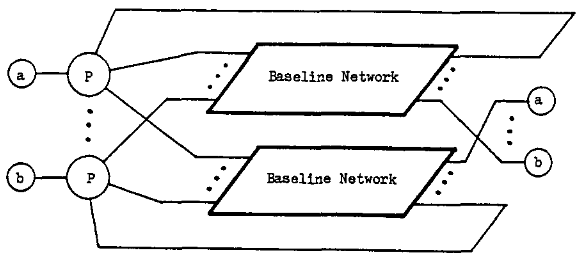

Fig. 3. Dual network configuration (P=processor).

Many communication strategies are possible other than the drop strategy or the hold strategy in the dual configuration. The following strategies are practical combinations and are studied in the later sections.

(1) Dual drop strategy
Two requests are generated to establish a path, one for each network. Each request acts the same as the request of the drop strategy in a single configuration. When a request establishes a path at a cycle, this information is passed to the sender at the end of the cycle. At the next cycle this information is sent to the other network as a release signal and the redundant request is discarded.

(2) Dual hold strategy
Two requests are generated to establish a path, one for each network. Each request acts the same as the request of the hold strategy in a single configuration. When a request establish a partial path up to the multiplexor stage at a cycle, this information is sent back to the sender at the end of the cycle. The request in the other network is discarded with a release signal from the sender at the next cycle. If both requests arrive at the multiplexor stage at the same cycle, one of them is selected by the priority set in the multiplexor.

(3) Single drop/dual drop strategy
Only one request is generated initially and submitted to a network chosen randomly. If this request is blocked, it is dropped and another request is generated in the other network at the next cycle. Thereafter the dropped request and newly generated request act the same as the requests of the dual drop strategy.

(4) Single drop/single drop strategy
Only one request is generated initially and submitted to a network chosen randomly. If this request is blocked, it is discarded and one

request is generated in the other network instead of the discarded request at the next cycle. This request acts the same as the previous request. If this request is blocked again, another request will replace this request in the other network, which the original request has been submitted to. This form of operation continues until one request finally establishes a path between the sender and the receiver.

(5) Single hold/dual hold strategy -

Only one request is generated initially and submitted to a network randomly chosen. If this request is blocked, the request holds the partial path already established. Besides this request, another request is generated in the other network and both requests act the same as the requests of the dual drop strategy thereafter.

III. General Assumptions and Regeneraton Model

A. Assumptions for analytical Models

Analytical models are built based on the following basic assumptions. When other assumptions are required , they will be described in the relevant section.

Assumption (1) When a path is established between a sender and a receiver, the path is used for a fixed number of cycles for data transfer. This data transfer time is denoted as 'd'. On the other hand, time between the completion of a request and generation of new request is variable and the requests are generated with rate 'r'. The rate of request is defined as the probability that a sender generates a new request per cycle given that the processor is idle.

Assumption (2): Destinations of the requests are distributed among N receivers with equal probability.

Assumption (3): If two requests arrive at a switching element at the same cycle and if they ask for a same output, one of them is randomly selected and passes the switching element while the other is blocked.

Assumption (1) shows the difference between service time distribution and request generation distribution. Assumption (2) is generally accepted in most performance models to simplify model building Assumption (3) is related to the design of a switching element and it is also generally adopted in performance models.

We classify the state of a sender (or a request generated by the sender) according to the stage number of the request and whether the request has been blocked in the network. The description of each state is as follows.

A_i(i=1..n) : The request is in stage i and it has not been blocked

B_i(i=1..n) : The request is in stage i and it has been blocked before.

C : Request has established a path and the path is currently used for data transfer.

D : Previous request generated by the sender has been completed and no new request is generated.

These are the basic state definitions. State B_i will be subdivided if necessary to denote the relationship between the blocked request and blocking request. The detail of the subdivision will be discussed in the hold model and drop model. We use the following conventions in describing the models. Transition probability from state S1 to S2 is written as P[S2|S1]. The absolute probability that a request is in state S1 is written as P(S1). To minimize the complexity of the drawing, state transition from a state to the state itself is not shown in the state transition diagram describing the model.

We can get the probability that a request is in a particular state when all the transition probabilities are expressed with the following three input parameters: network size, rate of request, and data transfer time.

B. Regeneration Model

We add one assumption which is also used in many previous models developed by other researchers. The purpose of this model is to illustrate the state transition diagram approach and to obtain the figures with which we compare results obtained from the hold and drop models.

Assumption (4) : If a request is blocked, it is discarded and another independent request is generated and submitted instead of the discarded one.

With assumption (4), there can be no B_i states in the transition diagram. The transition diagram of the regeneration model is relatively simple and shown in Fig.4.

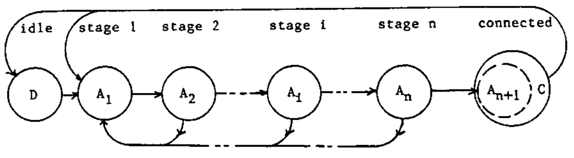

Fig. 4. State transition diagram of the regeneraton model.

Several transition probabilities are apparent and can be expressed as follows without further explanation.

$$P\{D|C\} = (1-r)/d \; , \qquad (1)$$

$$P\{A_1|C\} = r/d,$$

$$P\{A_1|D\} = r \; .$$

All the remaining transition probabilities can be obtained if we investigate a request in stage i. A request in stage i can pass the stage in three different ways as shown in Fig.2. The passing of the request is dependent on the state of other requests in the network and the probability that a request is in a specific state. If a request arrives at a switching element in stage i, the probability that one output is already occupied by other requests is the sum of the probabilities $P(A_{i+1})$ to $P(A_n)$ and P(C). This is the case (b) of Fig.2. Also the probability that another request arrives at the other input at the same cycle is

$P(A_i)$, which is the case (c) of Fig.2. The request may not encounter another request, as shown in case (a) of Fig.2, with remaining probability.

$$\text{Prob}\{\text{ case (a) of Fig.2 }\} = 1 - P(C) - \sum_{j=i}^{n} P(A_j), \quad (2)$$

$$\text{Prob}\{\text{ case (b) of Fig.2 }\} = \sum_{j=i+1}^{n} P(A_j) + P(C) ,$$

$$\text{Prob}\{\text{ case (c) of Fig.2 }\} = P(A_i) .$$

These probabilities are weighted with pass probabilities of each case shown in Fig.2 to get the transition probability from state A_i to state A_{i+1}. The request that failed to pass the stage changes its state to A_1. Transition to state A_{n+1} is considered as the transition to state C as shown in Fig.4.

$$P\{A_{i+1} | A_i\} \quad (3.1)$$
$$=\left(1-P(C)-\sum_{j=i}^{n} P(A_j)\right)+0.5\left(\sum_{j=i+1}^{n} P(A_j)+P(C)\right)+0.75\ P(A_i)$$
$$= 1 - 0.25\ P(A_i) - 0.5 \sum_{j=i+1}^{n} P(A_j) - 0.5\ P(C)$$

$$P\{A_1 | A_i\} = 1 - P\{A_{i+1} | A_i\} \quad (3.2)$$
$$= 0.25\ P(A_i) + 0.5 \sum_{j=i+1}^{n} P(A_j) + 0.5\ P(C)$$

All the transition probabilities are given in equations (1) to (3) and the model can be solved with an appropriate numerical method.

IV. Hold Model

In this model a blocked request holds the partial path which is already established. Assumption (4), which was used in the regeneration model no longer applies to this model and this model is built using only the assumptions (1) to (3). The general transition diagram of the hold model is shown in Fig.5.

A request in state A_i changes its state to B_i if it is blocked by another request. We call the latter request which prevented the pass of the former request as <u>blocking request</u> while we call the former request as <u>blocked request</u>. Since the transition from state B_i to A_{i+1} is related to the stage number of the blocking request, we subdivide the state B_i into B_i^j to denote the stage number of the blocking request. State B_i^j means that a request has been blocked and it is currently in

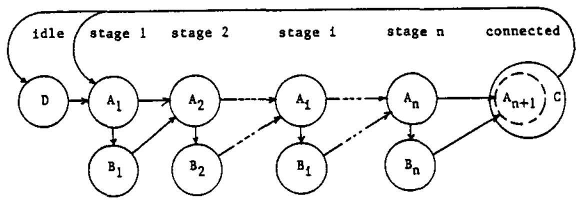

Fig. 5. State transition diagram of the hold model.

stage i and the blocking request is currently in stage j, where $j>i$.

As in the regeneration model, the following transition probabilities are apparent and shown without further explanation.

$$P\{D|C\} = (1-r)/d , \quad (4)$$
$$P\{A_1|C\} = r/d ,$$
$$P\{A_1|D\} = r .$$

The transition probability from state A_i to state A_{i+1} is similar to that of regeneration model. Instead of using A_i, we have to consider both A_i and B_i. However, the blocked request changes its state to B_i^j with superscript j to denote the stage number of the blocking request.

$$P\{A_{i+1}|A_i\} = \left[\ 1 - P(C) - \sum_{k=i}^{n} \left(P(A_k)+P(B_k)\right)\right] \quad (5.1)$$
$$= 1-0.25\left(P(A_i)+P(B_i)\right)-0.5\sum_{k=i+1}^{n} \left(P(A_k)+P(B_k)\right)-0.5P(C)$$

$$P\{B_i^j|A_i\} = \begin{cases} 0.25\ \left(P(A_i)+P(B_i)\right) & \text{for } j=i \\ 0.5\left(P(A_j)+P(B_j)\right) & \text{for } i<j<n \\ 0.5\ P(C) & \text{for } j=n+1 \end{cases} \quad (5.2)$$

When a request is in stage B_i, it cannot pass the stage until the blocking request completes the data transfer. Since the blocking request may progress down to the destination, the state of blocked request (B_i^j) is accordingly changed by changing the superscript j. Since a blocking request can be either in state A_j or in state B_j, to describe the pass probability of the blocking request we define the probability that a request in stage j can pass to stage j+1 with the following equation.

$$q_j = \text{Prob}\{\text{ A request in stage j passes the stage in a cycle }\} \quad (6)$$

$$= \frac{P(A_{j+1})}{P(A_j) + P(B_j)} \quad \text{for } 1<j<n$$

With this pass probability q_j, we can express the transition probabilities from state B_i^j as follows.

If the blocking request is in connected state, it will complete the data transfer with probability 1/d, and the blocked request will pass the stage with that probability. If the blocking request has not made a path yet, the blocked request cannot pass the stage and only the superscript value changes.

If a request is in state $B_i^j (j>i)$, the blocking request passes a stage with probability q_j, and the superscript value of the blocked request changes accordingly.

If a request is in state B_i^i, the blocking request has passed stage i at the previous cycle and it may pass another stage at the current cycle. Therefore the probability that a superscript changes its value by two is given as q_{i+1} while the probability that a superscript changes its value by one is $1 - q_{i+1}$. The transition probabilities are as follows.

$$P\{A_{i+1} | B_i^j\} = \begin{cases} 1/d & \text{for } j=n+1 \\ 0 & \text{for } j<n \end{cases} \qquad (7.1)$$

$$P\{B_i^{j+1} | B_i^j\} = \begin{cases} q_j & \text{for } i<j<n \\ 1 - q_{j+1} & \text{for } j=i \end{cases} \qquad (7.2)$$

$$P\{B_i^{j+2} | B_i^j\} = \begin{cases} 0 & \text{for } i<j<n \\ q_{j+1} & \text{for } j=i \end{cases} \qquad (7.3)$$

Equations (4) to (7) show all probabilities of the possible transition to other states. The state transition diagram can be solved by iterative substitution with any reasonable initial assignment of the probability to each state.

V. Drop Model

In this model a blocked request abandons the partial path which is already established and starts over again. Like the hold model, this model is built under the assumptions (1) to (3) and the following assumption.

Assumption (5) : Data transfer time is longer than the minimum time to establish a path between a sender and a receiver

The general transition diagram of drop model is shown in Fig 6. Definition of state A_i is the same as that of the regeneration model or the hold model. If a request is blocked at stage j, it changes its state to $B_{i,j}$. The first subscript i

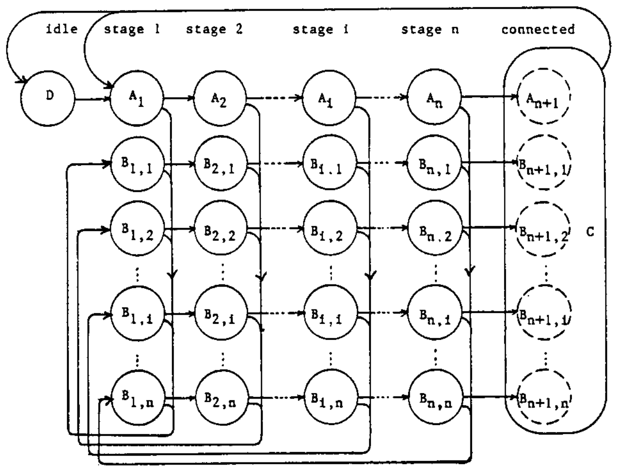

Fig. 6. State transition diagram of the drop model.

in state $B_{i,j}$ indicates the current stage of a request and the second subscript j indicates the stage where the request was most recently blocked. This request passes to stage 1 and 2 and so on until it reaches stage j if it is not blocked by other independent requests during the period. If the blocking request is still in the network without being blocked by other requests, the blocked request is blocked again by the same blocking request at stage j. To represent the stage number of the blocking request, state $B_{i,j}$ is subdivided to $B_{i,j}^k$ if necessary. When we say a request is in state $B_{i,j}^k$, it means a request is currently in stage i and it was blocked at stage j most recently and the blocking request was in stage k when the blockage occurred. We define one following term to simplify the written expression in the drop model.

$$S_i = \text{Prob}\{ \text{ A request is in stage i, where } 1<i<n \} \qquad (8)$$
$$= P(A_i) + \sum_{j=1}^{n} P(B_{i,j})$$

Like the previous regeneration and hold models, the following transition probabilities are easily obtained.

$$P\{D|C\} = (1-r)/d , \qquad (9)$$
$$P\{A_1|C\} = r/d ,$$
$$P\{A_1|D\} = r .$$

We describe the remaining transition probabilities, according to the current state of the request in following subsections.

A. Current state is A_i

A request in stage i can pass the stage in three different ways as shown in Fig.2. The probabilities of each case are as follows.

$$\text{prob\{ case (a) of Fig.2 \}} = 1 - P(C) - \sum_{j=i}^{n} S_j \,, \qquad (10)$$

$$\text{Prob\{ case (b) of Fig.2 \}} = \sum_{j=i+1}^{n} S_j + P(C) \,,$$

$$\text{Prob\{ case (c) of Fig.2 \}} = S_i \,.$$

The average pass probability from A_i to A_{i+1} is the weighted sum of probabilities shown in equation (10). The request that failed to pass the stage changes its state to $B_{i,i}^{k}$ where k indicates the stage number of the blocking request. The value of n+1 for superscript k indicates that the blocking request is in the connected state.

$$\begin{aligned} P\{A_{i+1} | A_i\} &= \Big[1-P(C)-\sum_{j=i}^{n} S_j\Big] \qquad (11.1)\\ &\quad + 0.5\Big[\sum_{j=i+1}^{n} S_j + P(C)\Big] + 0.75\, S_i \\ &= 1 - 0.25\, S_i - 0.5 \sum_{j=i+1}^{n} S_j - 0.5\, P(C) \end{aligned}$$

$$P\{B_{i,i}^{k} | A_i\} = \begin{cases} 0.25\, S_i & \text{for } k=i \\ 0.5\, S_k & \text{for } i<k<n \\ 0.5\, P(C) & \text{for } k=n+1 \end{cases} \qquad (11.2)$$

B. Current state is $B_{i,j}$ and $i \neq j$

This state represents a request that is currently in stage i and that was blocked most recently at stage j. Since the current stage number of the request is different from the stage number it was blocked recently, the request is independent of other requests in this stage and its transition probabilities to other states are similar to those from state A_i.

$$\begin{aligned} &P\{B_{i+1,j} | B_{i,j}\} \qquad (12.1)\\ &= \Big[1-P(C)-\sum_{j=i}^{n} S_j\Big] + 0.5\Big[\sum_{j=i+1}^{n} S_j + P(C)\Big] + 0.75\, P(C) \\ &= 1 - 0.25\, S_i - 0.5 \sum_{j=i+1}^{n} S_j - 0.5\, P(C) \end{aligned}$$

$$P\{B_{i,i}^{k} | B_{i,j}\} = \begin{cases} 0.25\, S_i & \text{for } k=i \\ 0.5\, S_k & \text{for } i<k<n \\ 0.5\, P(C) & \text{for } k=n+1 \end{cases} \qquad (12.2)$$

C. Current state is $B_{i,i}$, and $i \neq 1$

This state represents a request which returns to the previously blocked stage. If the blocking request still occupies the output of the switching element, the newly arrived request will be blocked again by the same blocking request. Since the blocking request itself can be blocked by other requests, we introduce following probabilities to describe the state change of the blocking requests.

q_k = Prob { A request in stage k passes the stage in a cycle}

$Q_{k,i}$ = Prob { A request 'R1' blocks request 'R2' again where 'R2' is previously blocked by 'R1' in stage i where 'R1' was in stage k when that blockage occurred. }

By definition,

$$q_k = \frac{P(A_{k+1}) + \sum_{j=1}^{n} P(B_{k+1,j})}{P(A_k) + \sum_{j=1}^{n} P(B_{k,j})} \quad \text{for } i<k<n \qquad (13)$$

Assumption (5) is used to remove the possibility that a newly connected request completes a data transfer. Without assumption (5), $Q_{k,i}$ becomes too complex to handle. $Q_{k,i}$ can be written as the products of q_j's and shown in equation (14).

$$Q_k = \begin{cases} \prod_{j=k}^{\min(n+1,k+i-1)} q_j & \text{for } k<n \\ 1-i/d & \text{for } k=n+1 \end{cases} \qquad (14)$$

Assume that a request was blocked at stage i by another request in stage k, where k>i. It takes i cycles for the blocked request to return to the stage i and the probability that the blocking request, which was in stage k, blocks the previously blocked request again is $Q_{k,i}$ by definition. If the blocking request is removed from the switching element in stage i either by dropping or completion of the data transfer, the blocked request is independent of other requests and can be treated as the request in state $B_{i,j}$ $(i \neq j)$. Special consideration must be given for those cases in which the blocking request was in stage i, i.e. the blocking request was randomly selected between two requests that arrived at the switching element at the same time. Since the blocking request has passed stage i with probability 1, the probability it occupies the switching element in stage i is $Q_{k+1,i-1}$ instead of $Q_{k,i}$.

$$P\{B_{i+1,i} | B_{i,i}^{k}\} \quad (15)$$

$$= \begin{cases} (1 - Q_{k,i})(1 - 0.25S_i - 0.5\sum_{j=i+1}^{n} S_j - 0.5\,P(C)) & \text{for } k \neq i \\ (1-Q_{k+1,i-1})(1-0.25S_i - 0.5\sum_{j=i+1}^{n} S_j - 0.5\,P(C)) & \text{for } k=i \end{cases}$$

Two different types of blockage are possible in stage i. One is the blockage by the same blocking request and the other is the blockage by the third independent request. These two probabilities are written separately in the following, but should be combined together in actual computation.

1. Case blocked by the same blocking request

$$P\{B_{1,i}^{j} | B_{i,i}^{k}\} = \begin{cases} Q_{k,i} & j=\min(k+1,n+1) \text{ for } k\neq i \\ Q_{k+1,i-1} & j=\min(k+i,n+1) \text{ for } k=i \end{cases} \quad (16)$$

2. case blocked by the third independent request

$$P\{B_{1,i}^{j} | B_{i,i}^{k}\} = \begin{cases} 0.25(1-Q_{k,i})S_i & j=i & \text{for } k\neq i \\ 0.5(1-Q_{k,i})S_j & i<j<n & \text{for } k\neq i \\ 0.5(1-Q_{k,i})P(C) & j=n+1 & \text{for } k\neq i \\ 0.25(1-Q_{k+1,i-1})S_i & j=i & \text{for } k=i \\ 0.5(1-Q_{k+i,i-1})S_j & i<j<n & \text{for } k=i \\ 0.5(1-Q_{k+1,i-1})P(C) & j=n+1 & \text{for } k=i \end{cases} \quad (17)$$

D. Current state is $B_{1,1}$

State $B_{1,1}$ is the special case of state $B_{i,i}$. This state is different from other $B_{i,i}(i \neq 1)$ states, because there is more chance of interference between blocking requests and blocked requests. Assume the following situation. A request 'R1' is blocked at stage 1 by request 'R2' which is in stage 3. Request 'R2' is blocked at stage 4 by a third request 'R3' at the next cycle. Then request 'R1' and 'R2' will compete at the switching element in stage 1 following that cycle because both requests have the same intended output By including this effect in drop model, the analytical model can provide more accurate results compared to the simulation result.

Consider first that the blocking request was in a connected state. If the blocking request completes the data transfer, then the blocked request normally passes stage 1. Otherwise it is blocked again and cannot pass stage 1. The conditional pass probability can be described as follows.

$$P(B_{2,1}^{n+1} | B_{1,1}^{n+1}\} = 1 - q_{n+1} \quad (18)$$

Next, if the blocking request was in between stage 2 and stage n, there are three cases to consider. The first case is that the blocking request passes a stage, second case is that the blocking request is dropped and wins the competition over the blocked request in stage 1. The third case is that the blocking request is dropped and loses the competition in stage 1. These cases are shown in following equation.

$$P\{B_{1,1}^{k+1} | B_{1,1}^{k}\} = q_k\ , \quad (19)$$

$$P\{B_{1,1}^{1} | B_{1,1}^{k}\} = 0.5\ (1-q_k)\ ,$$

$$P\{B_{2,1} | B_{1,1}^{k}\} = 0.5\ (1-q_k)\ .$$

If the blocking request was in stage 1, which means it was randomly selected between two requests arrived at the same cycle, the blocked request cannot pass stage 1 at the next cycle because the blocking request is not dropped at previous cycle. This is represented in following transition probability equation.

$$P\{B_{1,1}^{2} | B_{1,1}^{1}\} = 1 \quad (20)$$

All the transition probabilities are described in equations (9)-(20). In those equations, if the superscript of $B_{i,j}$ state is not mentioned, it applies to all the elements in that state equally. The superscript has actually no meaning if the first subscript value is larger than the second subscript value, which means that a blocked request has passed the stage it had been blocked. The solution of drop model can be obtained by iterative substitution like the hold model we already described.

VII. Simulation and Numerical Results

In this section, we estimate network performances by simulations. The simulation result is used to verify the result obtained from the analytical model. Each simulation is run for 80,000 node-cycles. That means 5,000 cycles for 16 senders or 10,000 cycles for 8 senders. To prevent the transient effect additional 100 cycles are run before actual data accumulation. The 80,000 cycles for a simulation run are divided into 10 trials. The results obtained from those 10 trials are averaged and standard deviation is computed to decide whether the obtained data represent the normal behavior of the network.

Average request service time is used as the primary measure of performance for both the analytical and simulation models. The request service time is defined as the time between the generation of a request and the completion of the request. In our analytical models, P(C) is the probability that a request spends time doing data transfer and P(D) is the probability that a sender is idle between the generation of two requests. Since the data transfer time is fixed and represented as 'd', average request service time can be written as follows

$$\text{Average request service time} = \frac{d\{1-P(D)\}}{P(C)} \quad (21)$$

First, the analytical model of a single network is compared to the simulation results. Since simulation is very costly, we limit the simulation to certain ranges of parameter values. We made the following selections: number of senders = 8, 16, 32, 64; rate of request = 0.1, 0.2, 1.0; data transfer time = 5, 10, 20. The result of the drop and hold strategies is shown in Table 1. In 36 cases we had only 2 cases in the drop model and 4 cases in the hold model that show a difference larger than 5 percent. The average differences between simulation and analytical model are 2.5 percent for drop model and 3.0 percent for hold model. The standard deviation of the request service time falls between 2 to 3 percent of average request service time in simulation results. In addition the drop strategy is shown to be better than the hold strategy except for very short data transfer time.

Next we test the significance of the assumption we removed from the drop model and the hold models. The result of the regeneration model, which include that particular assumption intentionally, are plotted with the results of our models in Fig.7. Simulation results are also shown in that figure for reference. We can see the difference of the request service time is as large as 30 percent in the case shown in Fig.7. There are some differences between the results obtained from the simulation and the regeneration model. One reason is that the regeneration model does not include the effects of the interference between blocking requests and it shows optimistic value for the request service time However, it serves as the ground work for more advanced models. The regeneration model differes from Patel's [2] model because it takes minimum n cycles to pass a network while it takes one cycle in Patel's model.

Table 1. Request service time in the single network configuration.

drop strategy

N	r	d=5			d=10			d=20		
		Simul.	Anal.	% diff	Simul.	Anal.	% diff	Simul.	Anal.	% diff
	1.0	14.82	14.42	-2.7	24.19	24.29	0.4	42.51	43.82	3.1
8	0.2	12.89	12.97	0.6	22.35	22.82	2.1	40.48	42.30	4.5
	0.1	11.38	11.72	3.0	20.44	21.31	4.3	38.98	40.69	4.4
	1.0	18.39	17.78	-3.3	28.83	28.63	-0.7	49.61	50.39	1.6
16	0.2	16.39	16.11	-1.7	26.54	27.11	2.1	47.75	48.71	2.0
	0.1	14.71	14.53	-1.2	24.45	25.35	3.7	44.91	46.91	4.5
	1.0	22.59	21.35	-5.5	34.17	33.25	-2.7	57.39	56.93	-0.9
32	0.2	20.30	19.49	-4.0	31.63	31.56	-0.2	55.37	55.11	-0.5
	0.1	18.01	17.63	-2.1	28.78	29.59	2.8	52.29	53.16	1.7
	1.0	26.74	25.14	-6.0	39.17	38.05	-2.9	65.84	63.55	-3.5
64	0.2	24.20	23.16	-4.3	37.23	36.24	-2.7	61.98	61.61	-0.6
	0.1	21.65	21.05	-2.8	33.81	34.09	0.8	60.34	59.55	-1.3

hold strategy

N	r	d=5			d=10			d=20		
		Simul.	Anal.	% diff	Simul.	Anal.	% diff	Simul.	Anal.	% diff
	1.0	14.53	14.04	-3.4	24.86	24.74	-0.5	45.37	46.23	1.9
8	0.2	12.43	12.61	1.4	22.41	23.09	3.0	43.12	44.42	3.0
	0.1	10.90	11.43	4.9	20.35	21.44	5.4	40.18	42.50	5.8
	1.0	17.87	17.16	-4.0	30.04	29.56	-1.6	52.92	54.50	3.0
16	0.2	15.83	15.50	-2.1	27.40	27.64	0.9	51.34	52.36	2.0
	0.1	13.72	14.00	2.0	24.77	25.66	4.9	49.53	50.15	1.3
	1.0	22.32	20.43	-8.5	35.80	34.47	-3.7	62.18	62.82	1.0
32	0.2	19.48	18.58	-4.6	32.59	32.33	0.8	58.37	60.37	3.4
	0.1	16.66	16.79	0.8	29.32	30.09	2.6	55.18	57.95	5.0
	1.0	25.77	23.87	-7.4	39.88	39.50	-1.0	72.46	71.18	-1.8
64	0.2	22.98	21.85	-4.9	38.25	37.22	-2.7	69.72	68.48	-1.8
	0.1	20.44	19.81	-3.1	35 62	34.73	-2.5	67.80	65.91	-2.8

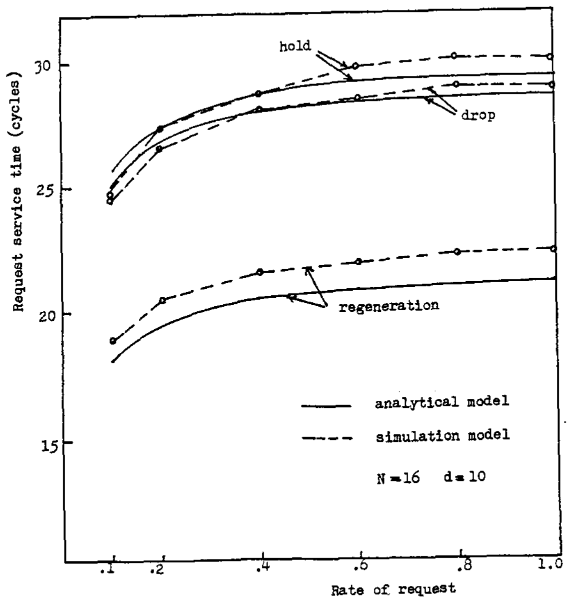

Fig. 7. Request service time in the single network configuration.

The performance of the dual configuration is obtained from simulation model only. Results of five different strategies described in section II are compared in Table 2.

The table is organized to show the effect of a specific parameter on the average request service time. The dual hold strategy shows best performance when data transfer time is shorter than 30 cycles. If the data transfer time is longer than 30 cycles, the dual drop strategy shows the best performance. Other parameters do not have much effect on deciding the best strategy. The performance of the single drop/single drop and single drop/dual drop strategies closely follows the performance of the dual drop strategy. Also the performance of the single hold/dual hold strategy is similar to that of the dual drop strategy. Therefore we can classify the five strategies into two groups; drop strategy and hold strategy as shown in Table 2.

The performance of the dual configuration is compared to the performance of the single configuration in Fig.8 varying the data transfer time. We use normalized request service time as a performance measure to show the result better. Normalization is done by dividing the average request service time by the minimum possible request service time in a single network configuration. As an example, we divided the request service time by 14 when d=10 and N=2^4. The dual configuration is better than the single configuration in its performance as expected. One noticeable thing in that figure is that the optimum

Table 2. Request service time in the dual network configuration.

N	d	r	DD	SD/SD	SD/DD	DH	SH/DH
			Group 1(drop)			Group 2(hold)	
16	10	0.1	20.45	20.66	20.93	19.54	19.86
16	10	0.2	22.52	22.27	22.37	21.17	21.64
16	10	0.5	23.48	23.41	23.46	22.08	22.16
16	10	1.0	23.74	24.08	24.12	22.49	22.70
16	5	1.0	15.51	15.59	15.51	14.03	14.17
16	10	1.0	23.74	24.08	24.12	22.49	22.70
16	15	1.0	32.97	32.79	31.98	31.07	31.50
16	20	1.0	41.55	41.25	41.22	38.69	40.80
16	30	1.0	58.39	57.84	57.04	57.55	58.14
16	40	1.0	71.95	72.58	75.48	78.63	80.80
8	10	1.0	21.42	21.36	21.84	20.65	20.64
16	10	1.0	23.74	24.08	24.12	22.49	22.70
32	10	1.0	26.10	26.42	26.80	25.25	25.49
64	10	1.0	29.00	29.53	29.56	27.40	28.01

DD:	Dual Drop
SD/SD:	Single Drop / Single Drop
SD/DD:	Single Drop / Dual Drop
DH:	Dual Drop
SP/DH:	Single Hold / Dual Hold

strategy in the dual configuration is different from the optimum strategy in the single configuration. We see that the drop strategy is better than the hold strategy in the single configuration. But in the dual configuration, we have to choose either the dual drop strategy or the dual hold strategy depending on the length of the data transfer time. A similar result for the single configuration and the dual configuration is shown in Fig.9 for different network sizes. We can see that the ratio of the performance level of the dual configuration to that of the single configuration becomes larger when the network size becomes larger. This gives some justification for the parallel network strategy especially in large networks.

VIII. Conclusions

We studied the performance of baseline networks in two configurations, single and dual. In the single configuration we built analytical models These models were compared to the regeneration model, which had been built used by other researchers. There are significant differences between the results obtained from the regeneration model and our models, which should not be ignored. With the two new models, we can better predict the actual performance level of the network. Also, we found that the drop strategy is better than the hold strategy in a single network except for short data transfer time.

For the dual configuration, we measured the performance level of the network with simulation. Five different strategies are proposed. Each strategy differs on the submission of the request to the network and type of action taken when the request is blocked. We found that the dual hold strategy works well when the data transfer time is relatively short and the dual drop strategy is better than the other strategies when the data transfer time is long. We also found that the dual network gives us good performance relative to the single network especially for large networks.

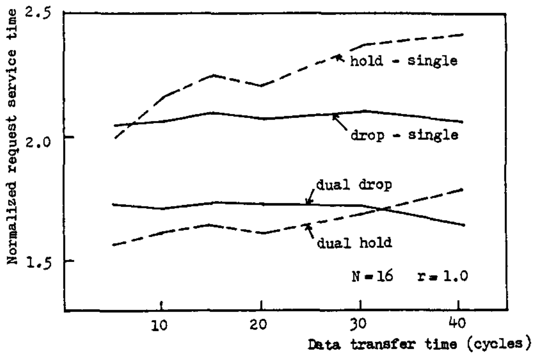

Fig. 8. Data transfer time vs normalized request service time.

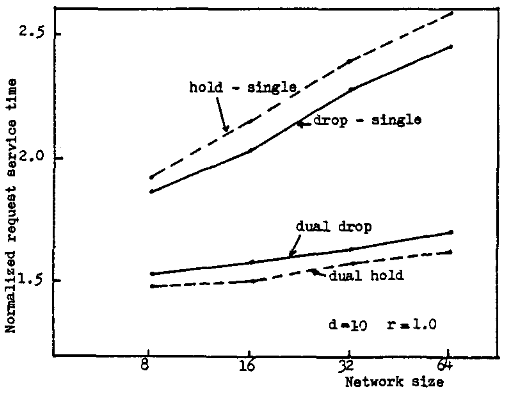

Fig. 9. Network size vs. normalized request service time

References

1. T. Feng, "A survey of Interconnection Networks," Computer, Dec. 1981, pp. 12-27.
2. J.H. Patel "Performance of Processor - Memory Interconnection for Multiprocessors," IEEE Trans. Comput. vol. C-30, Oct. 1981, pp. 771-780.
3 S. Thanawastien and V.P. Nelson "Interference Analysis of Shuffle/Exchange Network " IEEE Trans. Comput. vol. C-30, April 1981, pp. 545-556.
4. L.N. Bhuyan and C.W. Lee, "An Interference Analysis of Interconnection Networks," Proc. of the 1983 Inter. Conf. on Parallel Processing, Aug. 1983, pp.2-9.
5. D.M. Dias and J.R. Jump, "Analysis and Simulation of Buffered Delta Network," IEEE Trans. on Comput. vol. C-30, Aug. 1981, pp. 273-282.
6. C. Wu and T. Feng, "On a Class of Multistage Interconnection Networks " IEEE Trans. Comput. vol. C-29, Sep. 1980, pp. 694-702.
7. C. Wu, T. Feng and M. Lin, "Star: A local Network System for Real-Time Management of Imagery Data," IEEE Trans. Comput. vol. C-31, Oct. 1982, pp. 923-933.